ページング・マシンにおけるスワッピング・アルゴリズムの比較とプログラムの動作解析*

益田　隆司**　高橋　延匡**　吉沢　康文**

Abstract

This paper reports the comparison of swapping algorithms and some program behaviors under a paging environment. We developed the address trace system called PATTERN, which executes interpretively any program under HITAC 5020 Time Sharing System instruction by instruction, recording instruction code, instruction address and operand address for each instruction on the magnetic tape.

Some typical programs under HITAC 5020 TSS were traced using PATTERN. Some swapping algorithms were simulated using FORTRAN. We used those address trace tapes which are the outputs from PATTERN as inputs to the simulation, and found the relation among the mean step between page fault, memory size, page size and swapping algorithm.

We also found, based on the address trace tapes, the causal relation between the pages used in the past and those used in the future, which will give a useful information for pre-paging.

あ　ら　ま　し

virtual memory concept を有する計算機システムの解析を行なうためには，ページング機構のもとでのプログラムの動作解析が非常に重要である．プログラムの実行時のアドレス軌跡を，磁気テープ上にはきだすシステムを開発した．このシステムを PATTERN とよぶ．

セグメンテーション，ページング機構を有する HITAC 5020 タイムシェアリング・システムのもとで動作しているいくつかのプログラムのアドレス軌跡を PATTERN を用いて収集した．

本論文は，ページング機構下におけるいくつかのスワッピング・アルゴリズムを FORTRAN を用いてシミュレートし，PATTERN の出力であるプログラムのアドレス軌跡をそれに対する入力データとして，その解析結果をまとめたものである．それに関連した若干の解析データ，すなわち，プログラムのアドレス軌跡を適当な時刻できったときの，それまでに使用していたページと，それから将来に使用するページとの関係についての結果も報告する．

* The Comparison of Swapping Algorithms and some Program Behaviors under a Paging Environment, by Takashi MASUDA, Nobumasa TAKAHASHI and Yasufumi YOSHIZAWA (Central Research Laboratory, Hitachi Ltd.)

** 日立製作所中央研究所

1. ま　え　が　き

近年，virtual memory concept（以下，VMC と記す）を有する計算機システムが注目をあびている．VMC を有さないシステムでは，言語プロセッサはもとより，ユーザ・プログラムにおいても，各プログラムに割り当てられた主メモリ・サイズをこえたプログラムの場合には，各プログラムは，割り当てられた主メモリを時間的に多目的に用いて（overlay）いた．すなわち，各プログラムは，その大きさが割り当てられた主メモリの範囲をこえるときは，おのおのがその管理をしていた．

これに対して，VMC を有するシステムでは，主メモリと 2 次メモリのあいだの管理は，すべて，OS（Operating System）がつかさどる．そして，言語プロセッサ，あるいはユーザ・プログラムには，主メモリ・サイズよりもはるかに大きなアドレス空間が許される．このアドレス空間を論理アドレス空間（logical address space または，virtual address space）と呼ぶ．これに対して，物理的な主メモリ，2 次メモリに対応したアドレス空間を実アドレス空間（physical address space）と呼ぶ．論理アドレス空間，実アドレス空間内の各アドレスを，論理アドレス，実アドレスと呼ぶ．

命令実行の際に，論理アドレスから実アドレスへの

変換がなされる．変換された実アドレスが，主メモリ内のアドレスに対応していれば，そのまま実行される．実アドレスが，2次メモリ内のアドレスに対応していた場合には，2次メモリから主メモリへの情報の転送が行なわれる．この転送の管理は，OS が行なう．すなわち，実アドレスが2次メモリに対応していたときには，わりこみが発生する．

OS は，2次メモリから必要とされた語を含む1ブロック（ページと呼ばれる）を主メモリに転送するが，このとき，主メモリ内にあきページがなければ，主メモリ内はいずれかのページを2次メモリにおいだされなければならない．この際，おいだすべきページ選択のアルゴリズムが問題である．

また，転送の単位であるページ・サイズも問題である．ページ・サイズが小さいと，マッピング・テーブル（論理アドレスと実アドレスの対応を有するテーブル）が大きくなり，また，ページ・フォールト（実アドレスが2次メモリに対応する場合）の数も増大する．逆に，ページ・サイズが大きいと，ページの使用効率（ページが主メモリ内にスワップ・インされてから，スワップ・アウトされるまでのそのページの参照頻度）は低下すると思われる．

この論文では，HITAC 5020 タイムシェアリング・システム（5020 TSS）のもとで動作するいくつかのプログラムの実行時のアドレス軌跡をとり，それにもとずいて，いくつかのスワッピング・アルゴリズムの効率の比較を，ページ・サイズ，主メモリ・サイズなどをパラメータとして行なった結果について報告する．また，プログラムのアドレス軌跡を適当な時刻で切ったとき，その時点から過去に使用していたページと，そこから将来に使用するページの一致度を求めてみた．これは，pre-paging を行なう場合など，有効な情報である[6]．

2. 解析の方法

プログラムの実行時のアドレス軌跡を求め，その結果を磁気テープにはきだすシステムを開発した．このシステムを，PATTERN (Program Address Trace patTERN) と呼ぶ．

PATTERN により取り扱えるプログラムは，5020 TSS 下で動作する任意のプログラムである．解析の対象とするプログラムを 5020 TSS 下で動作するプログラムに選んだのは，それが，セグメンテーション，ページング機構を有するシステムであるため，そのもとで動作するプログラムは，物理的な主メモリ・サイズを考慮してかかれておらず，プログラムの構成が主メモリ・サイズの影響をうけることが，少ないと判断したからである．5020 TSS 下で動作するプログラムは，リエントラント形式になっており，プロシージャとデータあるいは作業用領域には，別々のセグメントが割り当てられ，プロシージャ・セグメントは，プログラムの実行に際し，何らの変更もうけない．これは 5020 TSS 下で動作するプログラムの特色である．

PATTERN を用いることにより，磁気テープ1巻に約60万ステップのアドレス軌跡が格納できる．各命令の磁気テープ上の型式を，**Fig. 1** に示す．磁気テープの型式は，FORTRAN で処理可能である．

PATTERN を用いる際，入力パラメータにより，以下のように出力量を制御できる．

（1） いくつかの指定したセグメントのアドレス軌跡

（2） OS だけのアドレス軌跡

（3） ユーザ・プログラムだけのアドレス軌跡

（4） 全実行命令のアドレス軌跡

などである．

Single Instruction

Instruction Code	α	β	
Procedure Base Register		Sequential Control Counter	
s_1		l_1	Double Instruction
s_2		l_2	
s_3		l_3	
s_4		l_4	

α: Bit of indirect addressing for the first operand
β: Bit of indirect addressing for the second operand
(s_1, l_1)=(operand segment, location)
(s_2, l_2)=(indirect segment, location)
(s_3, l_3)=(second operand segment, location)
(s_4, l_4)=(second indirect segment, location)

Fig. 1 Output format of PATTERN for each instruction.

3. 解析プログラムの種類と性質

解析の対象とするプログラムは，かなり多くのものを選定し，それらが全体として，TSS 下におけるプログラムを代表しているものであることが望ましい．

今回の解析で対象としたプログラムは，現在の 5020 TSS で代表的と考えられる以下の四つを選んだ．

おのおのについて簡単な説明を加える．

（1） BASIC によるコンパイル

BASIC は 5020 TSS における代表的な会話型言語である．インクリメンタル・コンパイル方式をとり，

1ステップのソース・プログラムごとにコンパイルが実行される.

選択したプログラムは，非線型直流回路解析用プログラムであり，663 ステップのソース・ステートメントからなる．ソース・プログラムは，すでにファイル・システムに登録されている．コンパイルの過程を磁気テープにはきだした．コンパイルに要した総ステップ数は，549,104 ステップである．使用セグメント数は，プロシージャ・セグメントが4個，おのおのに対応した作業用データ・セグメント (IDS: Internal Data Segment と呼ぶ) が4個，それ以外のデータ・セグメントが2個である．総ステップ数のうち，523,272 ステップがコンパイラ本体のセグメント内での処理である．

プログラム開始時より，約12,000ステップ経過後から，新しいページ要求は定常状態にはいっている．それ以降に要求される新しいページは，いずれもデータ・ページであり，ソース・プログラムを読みこみ，オブジェクト・プログラムをはき出すために必要となるデータ・ページである．

(2) PL/1W によるコンパイル

PL/1W は，システム記述用言語として開発された．PL/1 のサブセットよりなる．PL/1W コンパイラは，ソース・プログラムをマシン・インデペンデントな中間語に変換するフェイズ1，中間語をオブジェクト・プログラムに変換するフェイズ2からなる．5020 TSS 下で，PL/1W をはしらせると，約 220～250 ページ (256 語/ページ) 要求される．コンパイルを実行すると，通常，38個のセグメントが利用される．プロシージャ・セグメントは15個であり，IDS を除いた純粋のデータ・セグメントは8個である．

解析用に選んだソース・プログラムは，システム・プログラムの一部である“セグメント消去”をつかさどるモジュールであり，90ステップからなる．磁気テープ上のアドレス軌跡としては，後述の解析の際に，一つ一つのプログラムは機能が単純であるほうが望ましいと考え，フェイズ1だけを記録した．フェイズ1に要した全ステップ数は，356,172 ステップである．

プログラム開始時から約27,000ステップまでは初期状態であり，パラメータのとりこみ，分解，ネーム・テーブルへの登録，サーチ，組込み関数のネーム・テーブルへの登録などが行なわれる．27,000～130,000 ステップで，PROCEDURE 文，DECLARE 文のコンパイルが行なわれている．130,000 ステップ以降，一般のステートメントのコンパイルが行なわれている．DECLARE 文以外のステートメントは，69 ステップである．

(3) CONCISE によるシミュレーション・プログラムの実行

CONCISE は，5020 TSS 下で動作する会話型言語である．CONCISE コンパイラは，1行のソース・プログラムを読みとるごとに，インクリメンタルに，polish list の目的プログラムをつくりだし，それを解釈しながら実行していく．回帰関数の計算が可能であること，シミュレーションのための機能を備えていることがその特徴である．

使用セグメントは，プロシージャ・セグメントが6個，データ・セグメントが7個 (うち6個は IDS) である．また，CONCISE 本体のプロシージャの大きさは，2,700 ステップである．

今回の解析プログラムは，シミュレーションを行なっているものである．ページング・マシンにおいて，プログラムのページ間遷移の確率を与え，主メモリ・サイズ，スワッピング・アルゴリズムをかえて，シミュレーションを実行している．ソース・プログラムは70ステップからなる．アドレス軌跡の全ステップ数は，514,972 ステップである．新しいページ要求があるのは，100,000 ステップぐらいまでで，それ以降は新しいページ要求は全くない．

(4) 固有値を求める問題——EIGEN

プロシージャ中心の上記三つのプログラムに対して，データ参照中心のプログラムとして，対称行列の固有値，固有ベクトルを，JACOBI 法により求めるプログラムを選択した．くりかえし計算に際し，非対角要素の消去は，最大のものから行なうのではなく，逐次順々に行なっている．入力の行列の作成は次のようにした．入力として，α, β, γ を与える．α は対角要素ですべて等しい．非対角要素は $(0, \beta)$ の一様乱数として定める．γ は行列の次元である．これ以外に収束判定閾値 δ を与える．

今回の例題では，50次元の行列を取り扱った．計算途上で打ち切ったが，磁気テープ上の全ステップ数は，195,297 ステップである．

新しいページの要求があるのは，85,000ステップまでで，それ以降は，全く新しいページは要求されない．行列全体を対象としたくりかえし計算にはいってからは新しいページ要求はない．**Table 1** に，以上四つの各プログラムについての，全走行ステップ数，要求さ

Table 1 Programs to be analyzed.

	page size (word)	program size (page) proc.	data	sum	number of segments used proc.	data	sum	program steps (on Mag. Tape)
BASIC	64	44	547	591	5	8	13	549,105
	256	15	143	158				
	1,024	7	43	50				
PL/1W	64	148	194	342	12	19	31	356,173
	256	47	70	117				
	1,024	18	38	56				
CONCISE	64	56	67	123	6	7	13	514,972
	256	16	25	41				
	1,024	8	12	20				
EIGEN	64	16	171	187	5	5	10	195,296
	256	6	47	53				
	1,024	5	16	21				

れた全ページ数などのデータをしるしておく.

4. スワッピング・アルゴリズムの比較

前節の四つの解析プログラムを対象として,

スワッピング・アルゴリズム

主メモリ・サイズ

ページ・サイズ

とミッシング・ページ・フォールト間の走行ステップの関係を求めた. スワッピング・アルゴリズムを, FORTRAN を用いてシミュレートし, 各解析プログラムのアドレス軌跡を入力とする.

プログラム実行中, ページ・フォールトが生じた際, 主メモリ内にあきページが存在すれば, そこに要求されたページを読みこむ. あきページがないときには, すでに主メモリ内に存在するページのうちのいずれかを, 2次メモリにおいださなければならないが, そのページ選択の方式 (swapping algorithm) として, 以下の四通りを採用した.

(1) FIFO (First In First Out)

時間的に主メモリに最も長く存在しているページ, すなわち, 主メモリ内のページのうち, 時間的に最初に読みこまれたページをスワップ・アウトする.

(2) LRU (Least Recently Used)

スワップ・アウトの必要が生じたときから過去をみて, 時間的に最も遠い過去に参照された (近い過去には参照されていない) ページをスワップ・アウトする.

(3) RANDOM

主メモリ内のページをランダムに選択して, スワップ・アウトする.

(4) OPTIMUM[5]

スワップ・アウトする必要のある時点から将来をながめたとき, 現在主メモリ内に存在するページのうち, 最も遠い将来まで参照されないページをスワップ・アウトする.

ただし, (1), (3) の場合, 現在実行中の命令, あるいは, 現在実行中の命令が参照しているオペランドを含むページは, スワップ・アウトの対象から除く.

以上のアルゴリズムを適用するとき, スワップ・アウトは常に1ページずつ行なう. したがって, スワップ・インは要求に応じて (on demand) 完全に1ページずつである.

OPTIMUM は, スワップ・アウトすべきページの選択のために, 将来のプログラムの動きに関する情報を用いるので, 現実には実現不可能であるが, プログラムのアドレス軌跡があらかじめ磁気テープ上に格納されていれば, それをシミュレートしてみることは可能である. OPTIMUM からの結果は, 理論的最適値であり, その結果を実現可能な他の3つのアルゴリズムの結果と比較してみると興味深い.

ページ・サイズは, 16語, 64語, 256語, 1,024語の4種類を解析の対象とした (16語の場合は紙面の都合で省略する).

主メモリの大きさは, 各ページ・サイズに対して, 2ページからはじめ, 2のべき乗きざみで,

解析プログラムの使用ページ数 $< 2^i$ ページ

を満足する最小の i についてまで行なった. また, ページ・テーブルの影響は考慮していない.

結果を **Fig. 2** に示す. たて軸の msbf (mean step between page fault) は, ページ・フォールト間の実行ステップ数の平均である. 横軸に主メモリ・サイズをとり, スワッピング・アルゴリズム, ページ・サイズをパラメータとした. Fig. 2 では, 紙面の都合により, ページ・サイズ 256 語の場合のみをしるす.

msbf は, 主メモリがいっぱいになった時点からのアドレス軌跡のステップ数を, その間のページ・フォールトの数で除したものとして求めた. これは, 主メモリがいっぱいになるまでは, スワップ・アウトがおきないので, その影響を除くためである. ただこうして msbf を求めると, 各主メモリ・サイズごとに, 処理される解析プログラムのアドレス軌跡のステップ数が異なってくる. 解析プログラムの動作 (特に, 新しいページの要求など) が定常状態になっていれば, その影響は少ないが, 定常状態でないとあまりうまくない. そこで, 各解析プログラムについて, その動作が定常状態になっている範囲を求め, その下限と, 主メ

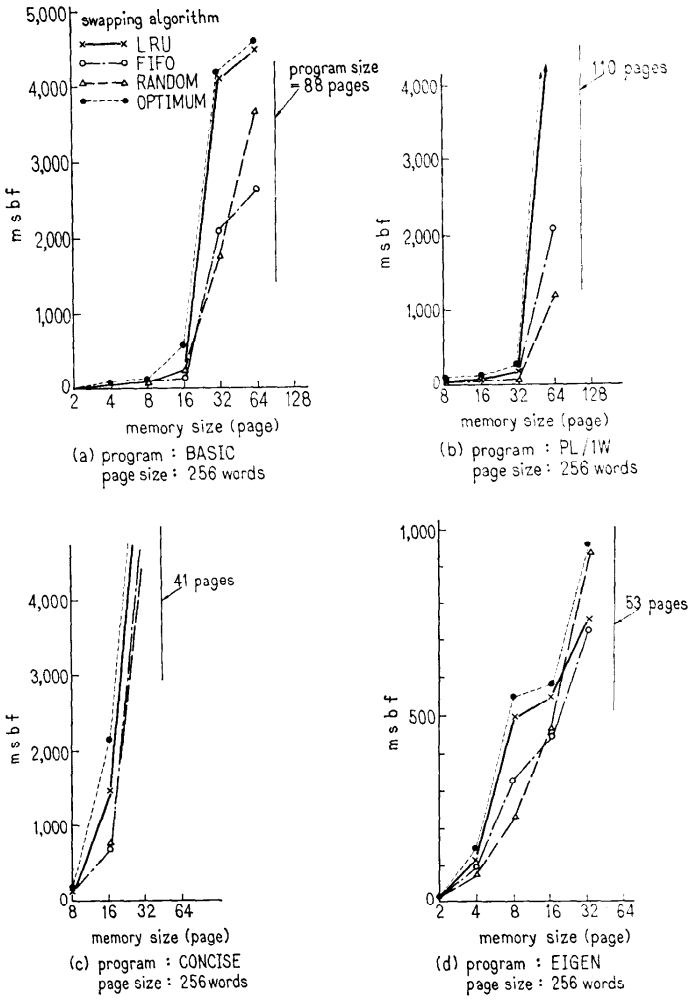

Fig. 2 msbf (mean step between page fault) vs. memory size, page size and swapping algorithm.

モリがいっぱいになる時点の時間的におそいほうを，アドレス軌跡から msbf を求める際の測定開始時点とした．Fig. 2 の結果を検討してみる．

（1） 主メモリ・サイズと msbf の関係

msbf は，主メモリ・ページ数がある一定のしきい値をこえると急速に大きくなっている．横軸が，2のべき乗の等間隔スケールであることを考慮してもそうである．

virtual memory 使用量に対して，どの程度の主メモリがあれば効率がよいかを，一応の目安として読みとった結果を，**Table 2** に示す．ページ・サイズ 256 語以外の場合についてもしるした．基準となる点の設定もしていないし，横軸のきざみもあらいため，さらにきめの細かい調査が必要である．

Table 2 で BASIC の場合の b/v が小さいのは，解析プログラムが，ソース・ステートメントのコンパイルをかなり早いループでくりかえしており，そのたびに使用されるデータ・ページの因果関係が弱いためである．

プログラムの型も無視した大雑把な近似では，あるプログラムを実行する際，そのプログラムが過去に要求したページの約50%が主メモリに存在すれば，msbf はかなり大きいといえる．この傾向は，スワッピング・アルゴリズムのいかんにかかわらず，大体一致している．

Table 2 をみると，いま一つの現象に気づく．BA-

Table 2 Memory size required for making msbf large.

c_s: memory size

a: c_s at which $\frac{d(\text{msbf})}{dc_s}$ becomes large.

b: c_s at which msbf is enough large.

	page size (word)	program size: v (page)	a	b	a/v	b/v
BASIC	64	314	32	64	0.10	0.20
	256	88	16	32	0.18	0.36
	1,024	32	8	16	0.25	0.50
PL/1W	64	307	64	256	0.20	0.83
	256	110	32	64	0.29	0.58
	1,024	56	16	32	0.28	0.57
CONCISE	64	123	32	64	0.26	0.52
	256	41	8	32	0.20	0.78
	1,024	20	4	16	0.20	0.80
EIGEN	64	187	64	128	0.34	0.68
	256	53	4	32	0.08	0.60
	1,024	21	4	16	0.19	0.76

SIC, CONCISE, EIGEN の場合は，ページ・サイズが大きくなるにしたがって，b/v の値が大きくなっている（効率が悪くなっている）のに対し，PL/1 W の場合には，b/v が小さくなっている（効率がよくなっている）．この原因は次のように考えられる．

PL/1 W の場合には，31個のセグメント（うち，プロシージャは12個）が，virtual memory 上に参照されているが，全アドレス軌跡の大半は，コンパイラ，トランスレータ本体のセグメント内での処理である．これはいずれも非常に大きなセグメントであり，このような大きなセグメント内での処理が中心のときには，ページ・サイズが大きいほうが効率がよい．

これに対して，ほかの三つの場合は，各プログラムで使用しているセグメントはかなり小さく（大半は，1,024語以下），そのセグメントのあいだをプログラムが頻繁に動く，といった処理が中心であり，この場合は，あきらかに，ページ・サイズ1,024は効率が悪くなる．

5020 TSS では，ページ・サイズ256語であり，また，セグメント間のリンクには完全なダイナミック・リンクの方式をとっている．各プログラムにおいて使用できるセグメントの数は全く自由である．したがって，数10語から成るセグメントも多い．ページ・サイズが1.024語の場合，こういった方式をとると当然効率は悪くなる．Table 2 の結果は，ページ・サイズが大きいときは，それに適したセグメントの利用方法を考えなければいけないことを示している．

Table 3 Comparison of swapping algorithm

	page size (word)	memory size (page)	msbf RANDOM (R)	FIFO (F)	LRU (L)	OPTIMUM (O)	R/O	F/O	L/O
BASIC	64	64	351	920	920	984	0.36	0.94	0.94
	256	32	1,790	2,117	4,188	4,188	0.43	0.51	1.00
	1,024	16	5,282	8,643	17,286	17,286	0.31	0.50	1.00
PL/1W	64	256	1,079	1,891	2,584	2,645	0.41	0.72	0.98
	256	64	1,232	2,076	5,645	5,750	0.21	0.36	0.98
	1,024	32	6,129	26,667	26,667	29,792	0.21	0.89	0.89
CONCISE	64	64	1,940	2,554	6,315	6,520	0.30	0.39	0.97
	256	32	8,906	19,296	28,944	28,944	0.31	0.66	0.66
	1,024	16	297,050	297,050	297,050	297,050	1.00	1.00	1.00
EIGEN	64	128	396	227	337	425	0.93	0.53	0.79
	256	32	941	728	758	950	0.99	0.77	0.80
	1,024	8	1,133	1,386	2,065	2,120	0.53	0.65	0.97

（2） スワッピング・アルゴリズムの比較

FIFO, LRU, RANDOM のうちでは，LRU が最もよいことが予想される．

解析結果からも，その傾向はうらずけられている．スワッピング・アルゴリズムの比較には，主メモリ・サイズが小さすぎても，大きすぎても好ましくないので，Table 2 の b，あるいは，それより一つ手前の主メモリ・サイズに対して，アルゴリズムの比較をしてみる．基準を OPTIMUM に選ぶ．結果を **Table 3** にまとめる．

総合的に判断すると，FIFO は RANDOM に比較して若干すぐれている程度である．主メモリ・サイズがある程度大きければ，FIFO が RANDOM に比較してすぐれている必然性はない．

EIGEN の場合には，RANDOM がよいという結果がでている．EIGEN は，ほかの三つのプログラムに比較すると，データ・エリアをひんぱんに，かつ，因果関係をあまりもたずに，ランダムに参照する傾向が強いプログラムである．このようなプログラムに対しては，FIFO はむろんのこと，LRU もあまりよい効果をもたないことを示している．一般に，メモリ参照に対しては，命令によるメモリ参照のほうが，データによるメモリ参照よりも，はるかに強い因果関係（過去に参照したページと将来に参照するページの一致度が高ければ，因果関係が強いということにする）を有する．LRU は因果関係の強いプログラムに対しては，すぐれた効果を示すはずである．Table 3 でもわかるように，PL/1W においては，LRU がすぐれた効果をもっているが，これは，PL/1 W がほかの三つに比較して，プロシージャの占める割合が多いためと判断される．

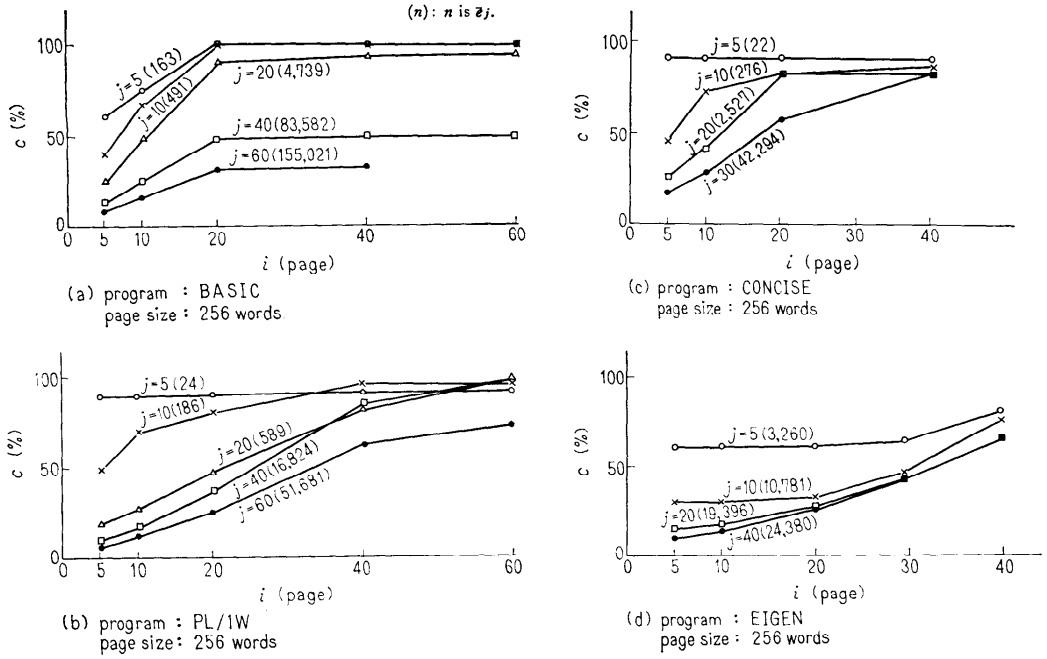

Fig. 3 The degree of coincidence between the pages used in the past and the pages used in future.

5. 過去使用ページと将来使用ページの関係[6]

プログラムの実行軌跡をいくつかの時点できり，そこから過去をながめたとき使用していたページと，将来に使用するページの一致度を求めてみた．これは，pre-paging を行なう場合など，有効な方法と考える．

プログラムのアドレス軌跡を，実行中の任意の点 $s_{(t)}$ できったとき，

(1) $s_{(t)}$ から過去をながめて，MRU (Most Recently Used) から，LRU にむけて，使用していた i 種類のページを，$p_{1(t)}, p_{2(t)}, \ldots, p_{i(t)}$ とする．以後，添字 t は，混乱しない限り省略する．

(2) $s_{(t)}$ から将来をながめたときに，順々に使用する j 種類のページを，$f_1, f_2, \ldots, f_j$ とする．

(3) $s_{(t)}$ から $f_{j(t)}$ を参照するまでの，アドレス軌跡のステップ数を，$e_{j(t)}$ とする．

(4) $(p_1, p_2, \ldots, p_i), (f_1, f_2, \ldots, f_j)$ について，相互で一致しているページの種類の組合せの数を a_{ij} とする．

(5) $A_{(t)}=(a_{ij(t)})$ とする．

今回の解析では，4つの解析プログラム共通に，

i=5, 10, 20, 40, 60 (若干の変更あり)

j=5, 10, 20, 40, 60 (〃 〃)

$s_{(1)}=50,000$, $s_{(2)}=100,000$, $s_{(3)}=150,000$, $s_{(4)}=200,000$

ページ・サイズ=64, 256, 1,024語に対して，$A_{(t)}$ を求めた．次に全体の平均として，

$$A=\sum_{t=1}^{4} A_{(t)}/4 \quad (\sum A_{(t)}=(\sum_t a_{ij(t)}))$$

$$\bar{e}_j=\sum_{t=1}^{4} e_{j(t)}/4$$

$$C=A/j\times 100$$

$$(A=(a_{ij});\ A/j\times 100=(a_{ij}/j\times 100))$$

を求めた．

各解析プログラムについて，ページ・サイズ，および，jをパラメータとし，C を Fig. 3 にしるした．Fig. 3 のたて軸 c は各 (i, j) に対し $a_{ij}/j\times 100$ として求めたものである．紙面の都合で，ページ・サイズ256語の場合のみしるす．同図の読みとり方を説明する．Fig. 3(b)，j=40 とする．40ページを要求するまでに走れる平均ステップ数，16,824ステップである．このとき，過去に使用していたページを40ページ保証しておけば，将来に使用する40ページのうち，約75% はその中に含まれることを意味している．j=40, i=60 でみると，将来に使用する40ページのうち，90% 近くが過去の 60 ページに含まれていることを示す．

Fig. 3 で，i を増加させると一致度はある値 α に収束している．プログラム実行にともない，ある時点以降で新しいページ要求がないときは，α は100%になる．新しいページの要求が常にある割合で存在する場合には，α はある定数に収束する．BASIC の場合，大きな j に対して，一致度が低いのは，プログラム実行に伴う新しいページ要求の割合が多いためである．

EIGEN の場合の一致度は，ほかの三つに比較して低い．たとえば，ページ・サイズ256語とすれば，Table 1 よりわかるように，53ページの virtual memory を使用する．Fig. 3 (d) より，$j=20$ の場合についてみると，$i=40$ としても，一致度は65%である．$i=40$ の場合は，$40/53 \simeq 0.8$ のごとく，全プログラム量の約8割が主メモリに存在しているにもかかわらず，一致度は非常に低い．EIGEN は行列を処理するプログラムであり，非連続的なデータ参照が多いプログラムである．LRU アルゴリズムは，この種の問題に対し，あまりすぐれていないことを示している．

6. むすび

セグメンテーション，ページング機構を有する HITAC 5020 タイムシェアリング・システムのもとで動作する任意のプログラムの実行時の全実行命令のアドレス軌跡を，磁気テープ上にはきだすシステム PATTERN を開発した．磁気テープ1巻に，約60万ステップのアドレス軌跡が格納でき，それは FORTRAN で処理可能である．

5020 TSS 下で動作しているプログラムから，代表的なものを四つ選び，PATTERN によりそのアドレス軌跡を求めた．それを用いて，ページング機構下のプログラムの動作に関するいくつかの基礎的なデータの収集・解析を行なった．すなわち，

(1) 主メモリ・サイズ，ページ・サイズ，スワッピング・アルゴリズムと msbf (mean step between page fault) との関係を把握．

(2) FIFO, LRU, RANDOM, OPTIMUM のスワッピング・アルゴリズムの比較．

(3) プログラムを実行中適当な時点できったとき，それまでに使用していたページと，それから将来に使用するページの一致度の比較．

などについて，定量的な解析結果を報告した．

解析用プログラムの選定，各種パラメータのきざみの選定など，今後さらに詳細な調査が必要である．

参考文献

1) 益田，他："セグメンテーション，ページング機構を有するタイムシェアリング・システムの評価"，情報処理第11回プログラミング・シンポジウム予稿集，pp. C 44～64 (1970)．
2) 本林，益田，勝枝，高橋："2次元番地付方式による HITAC 5020 TSS の特徴"，情報処理，Vol. 9, No. 6, pp. 317～325 (1968)．
3) 高橋，益田："HITAC 5020 TSS の解析と評価"，情報処理学会 OS シンポジウム報告集，pp. 6～36 (1970)．
4) Masuda, Yoshizawa, Hirosawa and Takahashi: "System Data and Its Evaluation of Time Sharing System with Virtual Memory Concept", Proc. of the MEXICO 1971 International IEEE Conference, pp. 8～11 (1971)．
5) L. A. Belady: "A Study of Replacement Algorithms for a Virtual Storage Computer", IBM System Journal, Vol. 5, No. 2, pp. 78～101 (1966)．
6) P. J. Denning: "The Working Set Model for Program Behavior", CACM, Vol. 11, No. 5, pp. 323～333 (1968)．

(昭和46年9月10日受付)